UNITED STATES PATENT APPLICATION

OF

**Ramesh PADMANABHAN, Pradeep SINDHU,
and Eric M. VERWILLOW**

FOR

SYSTEMS AND METHODS FOR GENERATING A RELIABLE CLOCK FOR RECEPTION AND RECOVERY OF DATA

SYSTEMS AND METHODS FOR GENERATING A RELIABLE CLOCK FOR RECEPTION AND RECOVERY OF DATA

BACKGROUND OF THE INVENTION

A. Field of the Invention

The present invention relates generally to communication systems and, more particularly, to systems and methods for generating a reliable clock to aid in the reception and recovery of data signals.

B. Description of Related Art

Some communication systems, such as synchronous optical networks (SONETs), transmit clock signals along with data signals. A system that receives the data signals may use the clock signals to recover the data signals. Sometimes, however, the received clock signals degrade or are lost during transmission. In this case, the receiving system cannot properly recover the accompanying data signals.

Therefore, there exists a need for a mechanism that facilitates the reception and recovery of data signals when unreliable clock signals accompany the data.

SUMMARY OF THE INVENTION

Systems and methods consistent with the present invention address this need by providing a reliable clock generator that converts from an unreliable clock domain to a fixed clock domain to facilitate the reception and recovery of data signals.

In accordance with the purpose of the invention as embodied and broadly described herein, a system for reliably receiving data includes a memory, write logic, and read logic. The

write logic receives data and an unreliable clock signal and writes the data to the memory using the unreliable clock signal. The read logic generates a gapped clock signal and reads the data from the memory using the gapped clock signal. The read logic generates the gapped clock signal by turning on and off a constant local clock signal.

In another implementation consistent with the present invention, a receiver includes a receiver component and a reliable clock generator. The reliable clock generator receives data and an unreliable clock signal and writes the data to a memory using the unreliable clock signal. The reliable clock generator also generates a reliable clock signal to compensate for underflow conditions in the memory, reads the data from the memory using the reliable clock signal, and provides the data and the reliable clock signal to the receiver component.

In a further implementation consistent with the present invention, a clock generator includes first and second state machines. The first state machine generates first and second enable signals. The first enable signal is used to read data from a memory that was written to the memory using an unreliable clock signal. The second state machine generates a gapped clock signal for reliably recovering the data in response to the second enable signal.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate an embodiment of the invention and, together with the description, explain the invention. In the drawings,

Fig. 1 is a diagram of an exemplary system in which systems and methods consistent with the present invention may be implemented;

Fig. 2 is a diagram of the reliable clock generator of Fig. 1 according to an implementation consistent with the present invention;

Fig. 3 is a detailed diagram of the reliable clock generator of Fig. 2 according to an implementation consistent with the present invention;

Fig. 4 is an exemplary diagram of the state machine of Fig. 3 according to an implementation consistent with the present invention;

Fig. 5 is an exemplary diagram of the first state machine of Fig. 4 according to an implementation consistent with the present invention;

Fig. 6 is an exemplary diagram of the second state machine of Fig. 4 according to an implementation consistent with the present invention; and

Figs. 7-10 are exemplary flowcharts of processing by the reliable clock generator of Fig. 1 according to an implementation consistent with the present invention.

DETAILED DESCRIPTION

The following detailed description of the invention refers to the accompanying drawings. The same reference numbers in different drawings identify the same or similar elements. Also, the following detailed description does not limit the invention. Instead, the scope of the invention is defined by the appended claims and equivalents.

Systems and methods consistent with the present invention provide a reliable clock generating mechanism that facilitates the reception of data when a potentially variable clock is received. The reliable clock generating mechanism converts from the potentially variable clock domain to a fixed local clock domain so that no data is lost under normal operating conditions.

Under abnormal clocking situations, such as those cases when the potentially variable clock is degraded or missing, the reliable clock generating mechanism provides a fixed clock to receiving logic to facilitate the reception and recovery of the data. Upon restoration of the potentially variable clock, the reliable clock generating mechanism permits the received data to track the potentially variable clock.

EXEMPLARY SYSTEM

Fig. 1 is an exemplary system 100 in which systems and methods consistent with the present invention may be implemented. The system 100 may be a receiver of a device, such as a network node, a switch, a computer device, etc., operating in a network, such as a SONET or a similar network. The system 100 may include a reliable clock generator 110 and a receiver component 120. The reliable clock generator 110 may receive data and an unreliable clock signal from a transmission medium, such as a network. The clock signal is considered unreliable because it may, in some instances, be degraded or missing. This may occur, for example, when failures exist in the network through which the clock signal is transmitted.

The reliable clock generator 110 generates a reliable clock signal to replace the unreliable clock signal and transmits the reliable clock signal, along with the data, to the receiver component 120. The receiver component 120 may include conventional receiver mechanisms to receive and recover the data using the reliable clock signal.

EXEMPLARY RELIABLE CLOCK GENERATOR

Fig. 2 is an exemplary diagram of the reliable clock generator 110 according to an implementation consistent with the present invention. The reliable clock generator 110 may include a memory 210, write logic 220, and read logic 230. The memory 210 may include a

first-in, first-out (FIFO) memory or a similar memory device. The write logic 220 receives data and an unreliable clock signal from a network, for example, and writes the data to the memory 210 using the unreliable clock signal. The read logic 230 generates a reliable clock signal and reads the data from the memory 210. The read logic 230 may then send the data from the memory 210 and the reliable clock signal to the receiver component 120 (Fig. 1).

Fig. 3 is a detailed diagram of the components of reliable clock generator 110 according to an implementation consistent with the present invention. As illustrated in Fig. 2, the clock generator 110 may include a memory 210, write logic 220, and read logic 230. In the implementation shown in Fig. 3, the memory 210 includes a FIFO 320. The FIFO 320 may include one or more conventional memory devices arranged as a FIFO that separates the clock domain of the write logic 220 from the clock domain of the read logic 230.

The write logic 220 receives data and an unreliable clock signal from a transmission medium, such as a network. In an implementation consistent with the present invention, the data includes SONET data (128 bits wide) and the unreliable clock signal includes a 77 MHz clock signal. The write logic 220 may include register 342, write pointer 344, logic 346, and register 348.

The register 342 may include a conventional memory device that buffers data for transmission to the FIFO 320 based on the unreliable clock signal. The write pointer 344 may include a conventional mechanism for generating an address for accessing the FIFO 320 to write data. The logic 346 may include conventional combinational logic that converts the write address generated by the write pointer 344 to a form for use by the read logic 230. For example, the logic 346 may convert the write address to a gray code sequence. The register 348 may

include a conventional memory device that buffers the gray code sequence (i.e., the write address) from the logic 346 using the unreliable clock signal.

The read logic 230 may use a constant clock signal to read the data from the memory 210 and generate a reliable clock signal for use by the receiver component 120. The read logic 230 may include register 362, multiplexer 364, registers 366, logic 368, register 370, comparator 372, gapped clock generator 374, read pointer 380, and multiplexer 382. The register 362 may include a conventional memory device that buffers data from the FIFO 320 in response to a data enable signal (DATA_EN) and a constant clock signal (CNSTCLK) of, for example, 157 MHz. The data enable signal (DATA_EN) assures that the register 362 reads data from the FIFO 320 only when data exists in the FIFO 320 and, thereby, prevents unstable data from being latched by the register 362.

The multiplexer 364 may include a conventional multiplexing device that receives the data received by the write logic 220 and the data buffered by the register 362 as inputs and outputs one of them as an output data signal. In an implementation consistent with the present invention, the multiplexer 364 normally selects the input from the register 362 except during a diagnostic mode.

The registers 366 may include conventional registers that act as a synchronizer to temporarily store and shift the gray code sequence (i.e., write address) from the write logic 220 in response to the constant clock signal (CNSTCLK). The logic 368 may include conventional combinational logic that converts the gray code sequence back into the write address. For example, the logic 368 may perform the opposite conversion as performed by the logic 346 to

restore the write address. The register 370 may include a conventional memory device that buffers the write address from the logic 368.

The comparator 372 may include a conventional mechanism capable of comparing two values and generating a result. The comparator 372 may compare the write address from the register 370 to a read address from the read pointer 380 to determine whether the FIFO 320 contains data. The comparator 372 sends the result of its comparison (i.e., a signal GT0 that indicates that there are "greater than 0" entries in the FIFO 320) to the gapped clock generator 374. The read pointer 380 may include a conventional mechanism for generating an address for accessing the FIFO 320 to read data.

The gapped clock generator 374 may receive the constant clock signal (CNSTCLK) and generate the data enable signal (DATA_EN) used by the register 362 and a gapped clock (GCLK) of, for example, approximately 78.5 MHz. The gapped clock generator 374 may include state machine 376 and register 378. The state machine 376 may include a finite state machine that uses the 157 MHz constant clock signal (CNSTCLK) to create an output that toggles every 157 MHz clock cycle, thereby creating a 78.5 MHz clock signal (i.e., a divide-by-two version of the 157 MHz clock). The state machine 376 creates the gapped clock (GCLK) by turning on and off the 78.5 MHz clock. The gapped clock signal (GCLK) compensates for underflow conditions in the FIFO 320 and aids in recovery of the data.

The register 378 may include a conventional memory device that stores the gapped clock signal (GCLK) generated by the state machine 376. The multiplexer 382 may include a conventional multiplexing device that receives the unreliable clock signal received by the write logic 220 and the gapped clock signal (GCLK) generated by the gapped clock generator 374 and

outputs one of them as a reliable clock signal. In an implementation consistent with the present invention, the multiplexer 382 normally selects the gapped clock signal except during a diagnostic mode.

EXEMPLARY STATE MACHINE

Fig. 4 is an exemplary diagram of the state machine 376 in an implementation consistent with the present invention. The state machine 376 may include two separate state machines 410 and 420 that operate in response to a reset signal. The reset signal occurs when data is received by the FIFO 320. The first state machine 410 may generate an enable signal (EN) to control operation of the second state machine 420. The second state machine 420 may generate a clock signal (GCLK) in response to the enable signal (EN) from the first state machine 410. The clock signal (GCLK) may be the gapped clock signal generated by the state machine 376.

Fig. 5 is an exemplary diagram of the first state machine 410 according to an implementation consistent with the present invention. The first state machine 410 may operate in three states 510-530. In this implementation, the state machine 410 changes state upon the occurrence of certain events. One of these events may include the receipt of the clock signal (GCLK) from the second state machine 420.

The state machine 410 enters the first state 510 upon receipt of the reset signal. In the first state 510, the state machine 410 generates the enable signal (EN) used by the second state machine 420 and a complement value for the data enable signal (!DATA_EN) used by the register 362 (Fig. 3). Upon receipt of the GT0 signal from the comparator 372, indicating that there is data in the FIFO 320, the state machine 410 enters the second state 520. In the second state 520, the state machine 410 generates the enable signal (EN) and the data enable signal

(DATA_EN). The state machine 410 remains in the second state 520 as long as the FIFO 320 contains data (i.e., as long as the comparator 372 generates the GT0 signal).

When no data remains in the FIFO 320, resulting in a complement value for the GT0 signal (!GT0), the state machine 410 enters the third state 530. In the third state 530, the state machine 410 may generate complement values for both the enable signal (!EN) and the data enable signal (!DATA_EN). Also, the state machine 410 may start a time-out counter upon entering the third state 530. The time-out counter times out (TO) when the FIFO 320 remains empty for a certain number of cycles. This may occur when the unreliable clock signal received by the write logic 220 (Fig. 1) degrades or disappears. The time-out period for the time-out counter should be long enough for all programmed inputs/outputs (PIOs) to complete successfully, such as 10 cycles.

As long as the FIFO 320 remains empty and the time-out counter has not timed out (!TO), the state machine 410 remains in the third state 530. If the time-out counter times out (TO), the state machine 410 returns to the first state 510, where it generates the enable signal (EN) and a complement value for the data enable signal (!DATA_EN). This transition may occur when the unreliable clock signal has not been received for longer than the time-out period.

If, on the other hand, the FIFO 320 receives data before the time-out counter times out (!TO), the state machine 410 returns to the second state 520, where it generates both the enable signal (EN) and the data enable signal (DATA_EN). This transition may occur when the write logic 220 begins to receive the unreliable clock signal again.

Fig. 6 is an exemplary diagram of the second state machine 420 according to an implementation consistent with the present invention. The second state machine 420 may

operate in two states 610 and 620. In this implementation, the state machine 420 changes state based on the value of the enable signal (EN) from the first state machine 410.

The state machine 420 enters the first state 610 upon receipt of the reset signal. In the first state 610, the state machine 420 remains idle. When the state machine 420 receives the enable signal (EN) from the first state machine 410, the state machine 420 enters the second state 620. Otherwise, the state machine 420 remains in the first state 610.

In the second state 620, the state machine 420 generates a constantly oscillating clock signal (GCLK) using, for example, the constant clock signal (CNSTCLK). The state machine 420 remains in the second state 620 until it receives a complement value for the enable signal (!EN) from the first state machine 410. When this occurs, the state machine 420 returns to the first state 610 and becomes idle, thereby discontinuing generation of the clock signal (GCLK). This may occur when the unreliable clock signal degrades or is missing.

Through the generation and non-generation of the clock signal (GCLK), the state machine 376 generates the gapped clock.

EXEMPLARY PROCESSING

Figs. 7-10 are exemplary flowcharts of processing by the reliable clock generator 110 according to an implementation consistent with the present invention. Processing begins when the write logic 220 (Fig. 3) receives data and possibly an unreliable clock signal from a transmission medium, such as a network. Once this occurs, one of at least four possible paths may be taken depending on the state of the unreliable clock signal. The unreliable clock signal may be in one of at least four different states: STATE 1 - the unreliable clock signal may exist and operate at the correct frequency; STATE 2 - the unreliable clock signal may exist and

operate at a frequency lower than the correct frequency; STATE 3 - the unreliable clock signal may exist and operate at a frequency higher than the correct frequency; and STATE 4 - the unreliable clock signal may be missing. Assume for this example that the "correct" frequency is 77 MHz.

STATE 1

When the unreliable clock signal exists and operates at the correct frequency, the processing of Fig. 7 may occur. The write logic 220 may write the received data to the FIFO 320 [step 710]. The write logic 220 may use the unreliable clock signal to write the data to the FIFO 320 at an address determined by the write pointer 344.

The read pointer 380 generates an address for reading the data from the FIFO 320. The comparator 372 compares the read address with the write address generated by the write pointer 344. When the comparison indicates that data exists in the FIFO 320, the comparator 372 generates signal GT0. As described above, the gapped clock generator 374 uses the signal GT0 and a constant clock signal of, for example, 157 MHz to generate the gapped clock (GCLK) and the data enable signal (DATA_EN) (i.e., enters second state 520 in Fig. 5) [step 720].

The gapped clock generator 374 may generate a clock that toggles every 157 MHz clock cycle, resulting in a 78.5 MHz clock. Using a read clock (i.e., the 78.5 MHz gapped clock signal) that is slightly faster than the write clock (i.e., the 77 MHz unreliable clock signal) guarantees that all of the data in the FIFO 320 is read out. A slightly higher read rate, however, implies that the FIFO 320 drain rate is higher than the fill rate and may result in a FIFO underflow condition. The gapped clock generator 374 compensates for underflow periods by putting gaps in the generated clock (GCLK) (i.e., switches between second state 520 and third

state 530 in Fig. 5) [step 730]. The gapped clock generator 374, through use of the data enable signal (DATA_EN), also prevents data from being read out of the FIFO 320 until new data exists in the FIFO 320.

Using the constant clock signal (CNSTCLK) and the data enable signal (DATA_EN), the register 362 reads data from the FIFO 320 [step 740]. The combination of the constant clock signal (CNSTCLK) and the data enable signal (DATA_EN) creates the read clock signal used by the register 362 to access the FIFO 320. The read logic 230 may provide the data from the FIFO 320 and the gapped clock (GCLK) to the receiver component 120 [step 750].

STATE 2

When the unreliable clock signal exists and operates at a frequency lower than the correct frequency, the processing of Fig. 8 may occur. The write logic 220 may write the received data to the FIFO 320 [step 810]. The write logic 220 may use the unreliable clock signal to write the data to the FIFO 320 at an address determined by the write pointer 344. Because the frequency of the unreliable clock signal is lower than the correct frequency, the write logic 220 writes the data to the FIFO 320 at a slower rate than in STATE 1.

The read pointer 380 generates an address for reading the data from the FIFO 320. The comparator 372 compares the read address with the write address generated by the write pointer 344. When the comparison indicates that data exists in the FIFO 320, the comparator 372 generates signal GT0. As described above, the gapped clock generator 374 uses the signal GT0 and a constant clock signal of, for example, 157 MHz to generate the gapped clock (GCLK) and the data enable signal (DATA_EN) (i.e., enters second state 520 in Fig. 5) [step 820].

The gapped clock generator 374 may generate a clock that toggles every 157 MHz clock cycle, resulting in a 78.5 MHz clock. In this case, the read clock (i.e., the 78.5 MHz gapped clock signal) is possibly much faster than the write clock (i.e., slower than the 77 MHz unreliable clock signal). The higher read rate implies that the FIFO 320 drain rate is higher than the fill rate and may result in more frequent FIFO underflow conditions. The gapped clock generator 374 compensates for underflow periods by putting longer gaps in the generated clock (GCLK) (i.e., remaining in third state 530 in Fig. 5 longer) [step 830]. The gapped clock generator 374, through use of the data enable signal (DATA_EN), also prevents data from being read out of the FIFO 320 until new data exists in the FIFO 320.

Using the constant clock signal (CNSTCLK) and the data enable signal (DATA_EN), the register 362 reads data from the FIFO 320 [step 840]. The read logic 230 may provide the data from the FIFO 320 and the gapped clock (GCLK) to the receiver component 120 [step 850].

<u>STATE 3</u>

When the unreliable clock signal exists and operates at a frequency higher than the correct frequency, the processing of Fig. 9 may occur. The write logic 220 may write the received data to the FIFO 320 [step 910]. The write logic 220 may use the unreliable clock signal to write the data to the FIFO 320 at an address determined by the write pointer 344. Because the frequency of the unreliable clock signal is higher than the correct frequency, the write logic 220 writes the data to the FIFO 320 at a faster rate than in STATE 1.

The read pointer 380 generates an address for reading the data from the FIFO 320. The comparator 372 compares the read address with the write address generated by the write pointer 344. When the comparison indicates that data exists in the FIFO 320, the comparator 372

generates signal GT0. As described above, the gapped clock generator 374 uses the signal GT0 and a constant clock signal of, for example, 157 MHz to generate the gapped clock (GCLK) and the data enable signal (DATA_EN) (i.e., enters second state 520 in Fig. 5) [step 920].

The gapped clock generator 374 may generate a clock that toggles every 157 MHz clock cycle, resulting in a 78.5 MHz clock. In this case, the read clock (i.e., the 78.5 MHz gapped clock signal) is possibly slower than the write clock (i.e., faster than the 77 MHz unreliable clock signal). The higher write rate implies that the FIFO 320 fill rate is higher than the drain rate and may result in a FIFO overflow condition. A FIFO overflow condition may cause the FIFO 320 to receive more data than it is capable of storing.

Using the constant clock signal (CNSTCLK) and the data enable signal (DATA_EN), the register 362 reads data from the FIFO 320 [step 930]. The read logic 230 may provide the data from the FIFO 320 and the gapped clock (GCLK) to the receiver component 120 [step 940]. If an overflow condition occurs, the reliable clock generator 110 may generate an error signal [step 950].

STATE 4

When the unreliable clock signal is missing, the processing of Fig. 10 may occur. Because there is no unreliable clock signal, the write logic 220 does not write any received data to the FIFO 320 [step 1010].

The read pointer 380 generates an address for reading data from the FIFO 320. The comparator 372 compares the read address with the write address generated by the write pointer 344. When the comparison indicates that data still exists in the FIFO 320, the comparator 372 generates signal GT0. As described above, the gapped clock generator 374 uses the signal GT0

and a constant clock signal of, for example, 157 MHz to generate the gapped clock (GCLK) and the data enable signal (DATA_EN) (i.e., enters second state 520 in Fig. 5) [step 1020].

In this case, the gapped clock generator 374 may generate a clock that toggles every 157 MHz clock cycle, resulting in a 78.5 MHz clock. Because the write clock does not exist, an underflow condition results. The gapped clock generator 374 stops generating the clock signal (GCLK) and starts a time-out counter (i.e., enters third state 530 in Fig. 5) [step 1030]. If the time-out counter times out (TO) before the write logic 220 begins receiving the unreliable clock signal again, the gapped clock generator 374 begins generating the clock signal (GCLK), but not the data enable signal (!DATA_EN) and waits for the unreliable clock signal to resume (i.e., enters first state 510 in Fig. 5) [step 1040]. If the unreliable clock signal resumes before the time-out counter times out (!TO), the gapped clock generator 374 begins generating the clock signal (GCLK) and the data enable signal (DATA_EN) to permit data to be read out of the FIFO 320 again [step 1050].

Thereafter, using the constant clock signal (CNSTCLK) and the data enable signal (DATA_EN), the register 362 reads data from the FIFO 320 [step 1060]. The read logic 230 may provide the data from the FIFO 320 and the gapped clock (GCLK) to the receiver component 120 [step 1070].

CONCLUSION

Systems and methods consistent with the present invention provide a reliable clock signal to aid in the reception and recovery of data when an unreliable clock signal accompanies the data. The systems and methods convert from the unreliable clock domain of the unreliable clock signal to the fixed clock domain of the reliable clock signal.

The foregoing description of preferred embodiments of the present invention provides illustration and description, but is not intended to be exhaustive or to limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from practice of the invention. For example, while a number of elements have been shown in Fig. 3, the functions of at least some of these elements may be implemented in software in other implementations consistent with the present invention.

Also, the unreliable clock signal has been described as a 77 MHz clock, the constant clock signal as a 157 MHz clock, and the gapped clock signal as a 78.5 MHz clock. In other implementations consistent with the present invention, the clock frequencies may be different.

The scope of the invention is defined by the claims and their equivalents.